PARAMETER VERIFICATION IN AN AUTHENTICATION SYSTEM AND METHOD

CROSS REFERENCE TO RELATED APPLICATIONS

This application is related to co-pending U.S. Patent Application entitled "Extensible Authentication System and Method", accorded serial number _____ filed on even date herewith under attorney docket number 10012156-1.

TECHNICAL FIELD

The present invention is generally related to the field of authentication and, more particularly, is related to a parameter sharing mechanism in an authentication system and method.

BACKGROUND OF THE INVENTION

Multifunction peripherals have become more and more common in the workplace and in the home. Typical multifunction peripherals often combine the functions of printing, faxing, scanning, and copying into a single device or system. Previously, offices may have had to purchase a printer for computers to print, a facsimile machine to transmit and receive faxes, a scanner to scan documents, and a copy machine to make copies. Now all of these functions are being combined in a single device to save workspace and provide significant cost savings and efficiency. In addition, some multifunction peripherals may provide digital sending capability that enables users to scan a document into digital form and then send the resulting digital document.

While a single multifunction peripheral may provide many different functions, it may not be the case that all individuals should have access to all of the capabilities thereof. In some cases, specific individuals may be provided with access to specific functions using some sort of security or authentication routine that limits access to specific functions to specified users. However, as more and more different functions are integrated into multifunction peripherals or any other machine that limits access to specific functions in a similar manner, the security

and/or authentication systems used with such devices have to be restructured in order to limit access to such new functionality. This results in inefficiency and additional cost to adapt previous security and/or authentication systems for new functions on a device of restricted use.

SUMMARY OF THE INVENTION

In light of the above, the present invention provides for a method for parameter authentication. The present method comprises the steps of examining an authenticated parameter table to determine whether an unauthenticated user parameter is listed therein, and, bypassing authentication of the unauthenticated user parameter if the unauthenticated user parameter is listed in the authenticated parameter table.

In another embodiment, a system for parameter authentication is provided. The system includes a processor circuit with a processor and a memory, and an authenticated parameter table stored in the memory. The system also includes authentication logic stored in the memory and executable by the processor. The authentication logic comprises logic that examines the authenticated parameter table to determine whether an unauthenticated user parameter is listed therein, and, logic that bypasses the authentication of the unauthenticated user parameter if the unauthenticated user parameter is listed in the authenticated parameter table.

In still another embodiment, the present invention provides for a program embodied in a computer readable medium for parameter authentication. In this respect, the program comprises code that generates an authenticated parameter table, code that examines the authenticated parameter table to determine whether an unauthenticated user parameter is listed therein, and, code that bypasses the authentication of the unauthenticated user parameter if the unauthenticated user parameter is listed in the authenticated parameter table.

Other features and advantages of the present invention will become apparent to a person with ordinary skill in the art in view of the following drawings and detailed description. It is intended that all such additional features and advantages be included herein within the scope of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention can be understood with reference to the following drawings. The components in the drawings are not necessarily to scale. Also, in the drawings, like reference numerals designate corresponding parts throughout the several views.

FIG. 1 is a drawing of a multifunction peripheral that employs an authentication system according to an embodiment of the present invention;

FIG. 2 is a block diagram of the authentication system employed in the multifunction peripheral of FIG. 1;

FIG. 3 is a block diagram of an authentication manager included in the authentication system of FIG. 2;

FIG. 4 is a block diagram of an authentication agent included in the authentication system of FIG. 2;

FIG. 5 is a flow chart of a "Request Authentication" method implemented in the authentication manager of FIG. 3;

FIG. 6 is a flow chart of an "Authenticate" method implemented in the authentication agent of FIG. 4;

FIG. 7 is a flow chart of a "Get" method implemented in the authentication manager of FIG. 3; and

FIG. 8 is a flow chart of a "Set" method implemented in the authentication manager of FIG. 3.

DETAILED DESCRIPTION OF THE INVENTION

With reference to FIG. 1, shown is a multi-function peripheral 103 according to an embodiment of the present invention. The multi-function peripheral 103 is a device that combines the operations of several other devices such as, for example, a copy machine, a scanner, a printer, a digital sender, and other devices. It may be the case that access to the various functions performed by the multi-function peripheral 103 is to be restricted to various individuals. The multi-function peripheral 103 includes an authentication system to verify that a user is whom they say they are in order to provide access to the functions of the multi-function peripheral 103 to which that individual is entitled.

In this respect, the multi-function peripheral 103 includes a processor 113 and a memory 116, both of which are coupled to a local interface 119. The local interface 119 may be, for example, a data bus with an accompanying control/address bus as can be appreciated by those with ordinary skill in the art. The processor 113, memory 116, and the local interface 119 make up a processor circuit that is generally known by those with ordinary skill in the art. The multi-function peripheral 103 may also include one or more display devices 123 and one or more user input devices 126. The display device 123 is coupled to the local interface 119 by virtue of the display interface 129. Correspondingly, the user input device 123 is coupled to the local interface 119 through one or more input interfaces 133. In this respect, the display interface 129 and the input interface 133 may comprise, for example, appropriate input/output cards or other such devices as are generally known by those with ordinary skill in the art.

The multifunction peripheral 103 may also include a number of components that are employed to perform the various functions of copying, scanning, printing, digital sending and other functions. Such components may include, for example, paper path hardware to guide paper during the performance of the various functions, a printing assembly, scanning sensors and other scanning hardware, copying hardware, and other components. Such components are generally known by those with ordinary skill in the art and not discussed herein in detail.

The input devices may comprise, for example, a keyboard, keypad, touch pad, touch screen, microphone, or one or more push buttons, *etc.* The display devices may comprises, for example, cathode ray tubes (CRTs), liquid crystal display screens, gas plasma-based flat panel displays, indicator lights, or other types of display devices, *etc.*

The multi-function peripheral 103 also includes various components that are stored on the memory 116 and are executable by the processor 113. These components include an operating system 136 and a multi-function control system 139. The multi-function control system 139 includes an authentication system 143 according to an aspect of the present invention.

The operating system 136 is executed to control the allocation and usage of hardware resources in the multifunction peripheral such as the memory, processing time and peripheral devices. In this manner, the operating system 136 serves as

the foundation on which applications depend as is generally known by those with ordinary skill in the art.

The multi-function control system 139 is executed by the processor 113 in order to control the various operations of the multi-function peripheral 103 in performing various functions including copying, scanning, printing, digital sending, and any other functions that may be formed by the multi-function peripheral 103. The authentication system 143 is implemented in the multi-function peripheral 103 to authenticate a user to ensure that they are who they represent themselves to be and to limit access to the user to the various functions of the multi-function peripheral 103 to which that individual is entitled. In one embodiment, the authentication system 143 is programmed in an appropriate computer language, such as, for example, C, C++, Java, and other appropriate programming languages.

The memory 116 is defined herein as both volatile and nonvolatile memory and data storage components. Volatile components are those that do not retain data values upon loss of power. Nonvolatile components are those that retain data upon a loss of power. Thus, the memory 116 may comprise, for example, random access memory (RAM), read-only memory (ROM), hard disk drives, floppy disks accessed via an associated floppy disk drive, compact discs accessed via a compact disc drive, magnetic tapes accessed via an appropriate tape drive, and/or other memory components, or a combination of any two or more of these memory components. In addition, the RAM may comprise, for example, static random access memory (SRAM), dynamic random access memory (DRAM), or magnetic random access memory (MRAM) and other such devices. The ROM may comprise, for example, a programmable read-only memory (PROM), an erasable programmable read-only memory (EPROM), an electrically erasable programmable read-only memory (EEPROM), or other like memory device.

In addition, the processor 113 may represent multiple processors and the memory 116 may represent multiple memories that operate in parallel. In such a case, the local interface 119 may be an appropriate network that facilitates communication between any two of the multiple processors, between any processor and any one of the memories, or between any two of the memories *etc.* The processor 113 may be electrical or optical in nature.

Next, a discussion is provided of the general operation of the multi-function peripheral 103 in authenticating a user to provide access to the various functions of the multi-function peripheral 103. Assume that a user approaches the multi-function peripheral 103 to perform one of the various functions provided thereby. For example, assume that a user wishes to scan in a document and transmit the same in a digital form to another individual using electronic mail. This may be done where the multifunction peripheral 103 is coupled to a network using a digital sending capability of the multifunction peripheral 103. Further assume that the individual must be authenticated as there are a limited number of people who have access to the digital sending capability of the multifunction peripheral.

To begin, the user manipulates the user input device 126 to indicate the particular function that the user desires to employ on the multi-function peripheral 103, such as the digital sending capability. In response, the multi-function control system 139 implements the authentication system 143 to verify that the user is the person that they claim to be in order to implement the appropriate function that they desire. The authentication system 143 thus displays a request on the display device 123 that the user enter appropriate user parameters such as, for example, a user password or user name, *etc.* Alternatively, the user may be required to provide user parameters such as biometric identification information or other identifying indicia through the various user input devices 126 employed with the multi-function peripheral 103.

Such user input devices 126 may comprise, for example, a keyboard to enter a user name or password, as well as more complex user input devices 126, such as a retinal scanner, fingerprint scanner, and/or other input devices. The user inputs the user parameters using one or more of the user input devices 126. The authentication system 143 then verifies the user parameters. For example, the parameters to be entered may be a user's password and user name. Such information may be stored in a network server that is coupled to the multi-function peripheral 103 through a local area network or other network. The authentication system 143 may authenticate the user parameters by requesting that the network server verify that the user name and password are active on the network if such status provides access to the digital sending capabilities of the multifunction peripheral. Alternatively, such information may be maintained in a database that is

consulted by the authentication system 143 to verify that the user has access to the digital sending capabilities.

With reference to FIG. 2, shown is a functional block diagram of the primary components of the authentication system 143 as it interacts with other components according to an embodiment of the present invention. As shown in FIG. 2, each block represents a module, object, or other grouping or encapsulation of underlying functionality as implemented in programming code. However, the same underlying functionality may exist in one or more modules, objects, or other groupings or encapsulations that differ from those shown in FIG. 2 without departing from the present invention as defined by the appended claims.

The authentication system 143 includes an authentication manager 153 and a number of authentication agents 156. According to an aspect of the present invention, the authentication system 143 can include any number of authentication agents 156. Communication between the authentication manager 153 and the authentication agents 156 is accomplished by sending appropriate calls and responses to each other as is customary in an object-oriented environment.

The authentication system 143 also includes an agent priority table 159. The agent priority table 159 is generated by the authentication manager 153 to track the existence of the authentication agents 156. The authentication system 143 also interfaces with the user input devices 126 and with external authentication services 163 to obtain authentication of user parameters as will be described. In addition, the authentication system 143 interfaces with an application 166 that is implemented upon successful authentication of a user. In the case of the multi-function peripheral 103 (FIG. 1), the application 166 may be a feature or function of the multi-function control system 139. For example, the application 166 might be the digital sending function, copy function, or print function, *etc.*

The authentication system 143 also includes a user parameter table 169. The user parameter table 169 is employed to store previously authenticated user parameters for later use by subsequent authentication agents 156. Specifically, if a prior authentication agent 156 has authenticated a user parameter, it is written to the user parameter table 169. All authentication agents 156 check the user parameter table 169 to see if a parameter that they are to authenticate has previously been authenticated. If such is the case, then the respective

authentication agent 156 skips the authentication task necessary to authenticate the previously authenticated user parameter.

Next, the operation of the authentication system 143 is described. To begin, upon startup of the multi-function peripheral 103, the authentication manager 153 executes a method to discover all of the authentication agents 156 that are stored in the memory 116. The discovery of the authentication agents 156 is accomplished, for example, using the Component Object Model (COM) created by Microsoft Corporation of Redmond, Washington. The COM architecture allows for the discovery of objects by employing a category manager with which objects are registered. For a discussion of the COM architecture, see Dale Rogerson, Inside COM, Microsoft Press, Redmond, Washington, 1997, the entire text of which is incorporated herein by reference.

The discovered authentication agents 156 are then listed in the agent priority table 159 that is ultimately consulted by the authentication manager 153 in sending authentication requests to the respective authentication agents 156. Each of the authentication agents 156 includes a priority level value. The order in which the authentication agents 156 are listed in the agent priority table 159 is based upon the priority level values within each of the authentication agents 156. Alternatively, the order in which the authentication agents 156 are listed in the agent priority table 159 may depend upon the order in which each of the authentication agents 156 was registered or included in the authentication system 143, or the order may be determined in some other manner.

The discovery of the authentication agents 156 in this manner points to the fact that the authentication system 143 is extensible in that there is no static coupling between the authentication manager 153 and the authentication agents 156. That is to say, the authentication manager 153 does not know how many authentication agents 156 there are until discovered. The fact that such static coupling does not exist in the architecture of the authentication agent 143 allows for the easy addition of authentication agents 156 thereto in specific applications without having to modify the authentication manager 153. Specifically, if a new type of authentication is desired for a particular function, all that is necessary is that a new authentication agent 156 be created and added to the authentication system 143, thus reducing time and effort to accomplish such a modification. The extensibility of the authentication system 143 is further reflected in other interaction

between the authentication manager 153 and the authentication agents 156 as will be discussed.

Each of the authentication agents 156 is configured to perform an authentication task that provides for the authentication of at least one user parameter that is supplied by a user by virtue of the user input devices 126. Each authentication agent 156 may authenticate one or more user parameters supplied by the user. The user parameters that are authenticated by a respective agent 156 may include at least one parameter that may or may not be unique with respect to the remaining authentication agents 156.

According to one aspect of the present invention, when a particular authentication agent 156 wishes to authenticate a parameter that was supplied by a user, then the authentication agent 156 may communicate with an appropriate external authentication service 163. Such an external authorization service 163 serves to authenticate the specific parameter. For example, the external authentication service 163 may reside in a server coupled to a local area network or other network. Assume that the multi-function peripheral 103 is also coupled to the same network. Such an external authentication service 163 may be employed, for example, able to verify that a specific username or password associated with a user that has access to a computer system on the network as can be appreciated by those with ordinary skill in the art.

Alternatively, the authentication agent 156 may include the functionality that causes the authentication of a particular parameter by itself without the use of an external authentication service 163. The actual act of authentication may involve, for example, comparing an unauthenticated parameter with a table or database of known parameters for a match. When a match occurs, the parameter has been authenticated. Note there are many other approaches that may be employed to authenticate parameters as is generally known to those with ordinary skill in the art.

Assuming that a user wishes to employ the multi-function peripheral 103 (FIG. 1) to perform a specific function such as transmitting a document by way of a digital sender, *etc.*, then the user indicates the desired function through an appropriate user input device 126. The authentication manager 153 then implements a "Request Authentication" method to obtain authentication of the user. Upon executing the Request Authentication method, the authentication manager

153 consults the agent priority table 159 to determine the first authentication agent 156 that is to be called upon to authenticate a particular parameter of the user.

The authentication manager 153 then sends a message to such authentication agent 156 that includes the desired function that the user wishes to implement on the multifunction peripheral 103. Upon being called by the authentication manager 153 to authenticate the user, the first authentication agent 156 determines whether it is supposed to authenticate the user for the function that the user wishes to implement. That is to say, a particular authentication agent 156 may or may not be used to authenticate a user for a predetermined function of the multi-function peripheral 103.

Assuming that the authentication agent 156 does perform an authentication procedure for the desired function, then the authentication agent 156 executes a method to query the user to enter or otherwise provide a user parameter for authentication. Upon obtaining the user parameter, the authentication agent 156 then requests the authentication manager 153 to determine whether the specific user parameter to be authenticated is listed in the user parameter table 169 and the authentication manager 153 examines the user parameter table 169 and responds to the authentication agent 156 accordingly. If the user parameter is listed in the user parameter table 169, then it is not necessary to authenticate the specific user parameter as it was previously authenticated. In such a situation, the authentication agent 156 responds to the authentication manager 153 that the user parameter has been authenticated. On the other hand, if the user parameter is not listed in the user parameter table 169, then the authentication agent 156 proceeds with the authentication of the user parameter.

Assuming the user parameter is authenticated, the authentication agent 156 returns a "valid" message to the authentication manager 153 indicating that the user was authenticated. The authentication manager 153 then moves to obtain authentication from the next authentication agent 156, if there are any remaining to query. If the authentication was unsuccessful, then the authentication agent 156 returns a "rejected" message indicating that the user has not been authenticated. In such case, the authentication manager 153 responds by denying the user access to the desired function of the multifunction peripheral 103 or other device.

If it is the case that the authentication agent 156 does not perform authentication for the desired function, then the authentication agent 156 returns a

"valid" message to the authentication manager 153 indicating that the user has been authenticated. This action provides further evidence of the extensibility of the architecture of the authentication system 143. In particular, if an authentication agent 156 is not to perform an authentication task for a specific desired function to be accessed, then by sending a "valid" reply, the authentication manager assumes that the user was authenticated by the specific authentication agent 156 an proceeds accordingly. Alternatively, a separate message may be sent to the authentication manager 153 by the authentication agent 156 that indicates that the authentication agent 156 does not perform authentication for the desired function. In such case, the authentication manager 153 should be configured to recognize such a message. In any event, the authentication manager 153 would proceed with the authentication procedure regardless of whether a "valid" message or an "inapplicable" message is received from the authentication agent 156.

To authenticate a user, an authentication agent 156 first obtains the user parameter through an appropriate user input device 126. Thereafter, the authentication agent 156 either performs the authentication of the user parameter itself or requests an external authentication service 163 to authenticate the user parameter.

In those situations where the user parameters are actually authenticated by the authentication agent 156, then the authentication agent 156 provides the newly authenticated user parameter to the authentication manager 153 in a request that the authentication manager 153 write the same to the user parameter table 169. Such a request may be, for example, a call to the authentication manager 153 that identifies a "Set" method that is to be executed with the user parameter that writes the same to the user parameter table 169.

In implementing the Request Authentication method, the authentication manager 153 sends an authentication request to each one of the authentication agents 156 based upon the position in the agent priority table 159. Upon receiving a response that indicates that the authentication was a success from any one of the authentication agents 156, then the authentication manager 153 proceeds to request authentication from the next authentication agent 156 listed in the agent priority table 159 until the last authentication agent 156 is queried. When the last authentication agent 156 has indicated that the user has been authenticated whether the authentication agent 156 performs an authentication task or is

bypassed, then the authentication manager 153 returns the final authentication result (i.e. passed or failed) to the multifunction control system 139. The multifunction control system 139 then either allows or prevents the desired function in the multi-function peripheral 153 that the user wishes to access.

Turning to FIGS. 3 and 4, shown are block diagrams of the authentication manager 153 and the authentication agent 156 according to an aspect of the present invention. In this respect, the authentication manager 153 includes a "Discover" method 173, a "Request Authentication" method 176, a "Get Attribute" method 179, and a "Set Attribute" method 183. The Discover method 173 is executed in order to discover all authentication agents 156 that are stored in the memory 116 as discussed above. The Request Authentication method 176 is employed to authenticate a user as described above. The Get Parameter and Set Parameter methods 179 and 183 are executed to examine and obtain user parameters from the user parameter table 169 and to write newly obtained user parameters thereto.

The authentication agent 156 includes an "authenticate" method 186 that is implemented to authenticate the user. Also associated with the authentication agent 156 are function variables 189, a priority level variable 193, and authentication parameters 196. The function variables 189 identify those functions of the multi-function peripheral 103 or other device for which the authentication agent 156 implements the Authenticate method 186. The priority level variable 193 provides a benchmark by which the authentication agent 156 is to be listed in the agent priority table 159 (FIG. 2). The authentication parameters 196 identify those parameters for which the respective authentication agent 156 performs its respective authentication tasks.

With reference to FIG. 5, shown is a flowchart of the Request Authentication method 176 according to an aspect of the present invention. Alternatively, the flowchart of FIG. 5 may be viewed as depicting steps in a method implemented in the multi-function peripheral 103 in authenticating a user.

Beginning with box 206, the first authentication agent 156 (FIG. 2) to which an authentication request is to be sent is looked up in the agent priority table 159 (FIG. 2). Thereafter, in box 209 the Request Authentication method 176 sends an authentication request to the authentication agent 156 identified in box 206. Then, in box 213 the Request Authentication method 176 determines whether it has

received a response from the authentication agent 156 (FIG. 2). If not, then the request authentication 176 proceeds to box 216 in which it is determines whether a time-out period has tolled.

The time-out is a predefined value that is stored in the memory 116. The authentication system 143 includes a timer that tracks whether or not the time that the authentication manager 153 waits for a response from the respective authentication agent 156 has gone beyond the predefined time-out. If there is no time-out in block 216, then the Request Authentication method 176 reverts back to box 213.

If a response is received from the authentication agent 156 in box 213, then the Request Authentication method 176 proceeds to box 219. In box 219, the Request Authentication method 176 determines whether the response from the authentication agent 156 indicates whether the user has been authenticated or whether the user was not successfully authenticated. Specifically, such response may indicate that the user is "valid" or "rejected", *etc.* If the authentication was unsuccessful, then the Request Authentication method 176 proceeds to box 223. The Request Authentication method 176 also proceeds to box 223 upon an occurrence of a time-out in box 216. In box 223 an indication of an authentication failure is provided to the user through an appropriate display device 123 (FIG. 1) and the user is denied access to the desired function of the multifunction peripheral 103. Thereafter, the Request Authentication method 176 ends as shown.

With reference back to box 219, if the response from the authentication agent 156 indicates that the authentication of the user was successful, then the Request Authentication method 176 proceeds to box 226 in which it is determined whether authentication agents 156 remain in the agent priority table 159 to which an authentication request has not been sent. If so, then the Request Authentication method 176 moves to box 229 in which the next authentication agent 156 is looked up in the agent priority table 159. Thereafter, the Request Authentication method 176 reverts back to box 209 to interface with the next authentication agent 156 to perform the next authentication task. With reference back to box 226, if the last authentication agent 156 has performed its authentication task as requested by the Request Authentication method 176, then the Request Authentication method 176 proceeds to box 233 in which an appropriate application 166 is implemented that enables the desired function within the multi-function peripheral 103 for use by the

user. It is understood that the authentication system 143 (FIG. 2) may be employed with any appropriate system or device and that the multifunction peripheral 103 is discussed herein merely to provide an example of the use of the authentication system 143.

Thereafter, in box 235, the Request Authentication method 176 determines whether there is another function to be performed. This may be determined, for example, by receiving a user input indicating a desire to perform another function. Alternatively, the Request Authentication method 176 may automatically assume that no further function is to be performed after a time out period of inactivity that occurs after a user has stopped using the multifunction peripheral 103. If more functions are to be performed in box 235, then the Request Authentication method 176 reverts back to box 206. Otherwise, the Request Authentication method 176 proceeds to box 236. In block 236, the Request Authentication method 176 resets the user parameter table 169 by erasing any values stored therein. Then, the Request Authentication method 176 ends.

With reference to FIG. 6, shown is a flowchart of the Authenticate method 186 according to another aspect of the present invention. Alternatively, the flowchart of FIG. 6 may be viewed as depicting steps in a method implemented in the multi-function peripheral 103 according to the present invention. The Authenticate method 186 is implemented within the authentication agent 156 (FIG. 2) in order to perform or broker the performance of an authentication task as requested by the authentication manager 153 (FIG. 2).

Beginning with box 250, the Authenticate method 186 first receives the authentication request from the authentication manager 153. The authentication request identifies the function that a user wishes to employ in the multi-function peripheral 103 for which the user is to be authenticated. The function may be identified, for example, as an attribute in the request. In box 250 the Authenticate method 186 compares the function in the request with the function variables 189 (FIG. 4) associated with itself. Then, in box 253, if a match between the supplied function and one of the function variables 189 is not detected, then the Authenticate method 186 proceeds to box 256 in which a message is sent back to the authentication manager 153 that indicates that the authentication was successful. This is done even though there was no authentication performed since the authentication agent 156 does not perform authentication services for the particular

function in question as was described above. Alternatively, a response may be sent to the authentication manager 153 that the authentication agent 156 is not applicable to the specified function. In such case, the authentication manager 153 assumes that the authentication was successful and proceeds to request authentication from the remaining authentication agents 156 as described previously.

With reference back to box 253, if there is a match detected between the function in the request and the function variables 189 associated with the authentication agent 156, then the Authenticate method 186 proceeds to box 259 in which a request is generated and sent to the authentication manager 153 to check the user parameter table 169 to determine whether the user parameter(s) has or have been previously authenticated by another authentication agent 156. Such a request may be, for example, a call to the authentication manager 153 to execute the Get Parameter method 179 (FIG. 3) with the user parameter supplied as an attribute, *etc.* If multiple parameters are to be compared with those user parameters stored in the user parameter table 169, then multiple calls may be made to the authentication manager 153, one for each unauthenticated user parameter.

Thereafter, in box 263, the Authenticate method 186 determines if any or all of the user parameters have previously been authenticated and were listed in the user parameter table 169. This is determined from a response from the authentication manager 153 that is generated by the execution of the Get Parameter method 179 that examines the user parameter table 169. For example, if the response returns the user parameter, then it is assumed that the parameter has been previously authenticated. If the parameter was not found in the user parameter table 169, then the response includes an empty parameter and it is assumed that the user parameter had not been previously authenticated.

If in box 263 all of the one or more user parameters in question were previously authenticated, then the Authenticate method 186 proceeds to box 256 in which a message is sent to the authentication manager 153 that the one or more user parameters were successfully authenticated. In this manner, the Authenticate method 186 bypasses the authentication tasks of these user parameters since they had previously been authenticated as evidenced by being listed in the user parameter table 169.

If at least one of the one or more user parameters in question were not previously authenticated, then the Authenticate method 186 proceeds to box 266 in which user parameters are obtained from the user through an appropriate user input device 126. In this respect, the authentication agent 156 may implement appropriate methods or logic that generate various input interfaces as can be appreciated by those with ordinary skill in the art. The various user parameters that may be input into the authentication system 143 for verification by the various authentication agents 156 may include, for example, a user password, a pin number, a user name, biometric information such as, fingerprints, retinal scans, voice scans, DNA information, smart card parameters, ID card parameters, and other quantifiable information.

Next, the Authenticate method proceeds to box 269 to obtain authentication of the unauthenticated user parameters from an external authentication service 163 (FIG. 2). Alternatively, in box 269, the Authentication Agent 156 may perform the authentication function itself by the execution of one or more methods, *etc.* The authentication may entail, for example, comparing the user parameter received from the user with a known list or database of user parameters to determine whether a user is listed as having privileges to perform various functions on the multi-function peripheral 103 or other device. Note that those user parameters that were previously authenticated and were included in the user parameter table 169 are not authenticated in box 169 as such action would be redundant. This results in faster authentication of the user parameters.

In box 266, the Authenticate method 186 determines whether the user parameters have been successfully authenticated. If not, then in box 269 the Authenticate method 186 returns a message to the authentication manager 153 informing the manager that the authentication was unsuccessful. Such a response may include an "invalid" indicator or other indication. Thereafter, the Authenticate method 186 ends accordingly.

On the other hand, if the authentication was deemed successful in box 266, then the Authenticate method 186 proceeds to box 279 in which the newly authenticated user parameters are placed in the user parameter table 169. In doing so, the Authenticate method 186 may call the Set Parameter method 183 supplying the newly authenticated user parameters as attributes, *etc.* Thereafter, the Authenticate method 186 proceeds to box 256 in which a message is returned

to the authentication manager 153 indicating that the authentication of the user parameter was successful. Thereafter, the Authenticate method 186 ends accordingly.

Turning next to FIG. 7, shown is a flow chart of the Get Parameter method 179 according to an aspect of the present invention. Alternatively, the flowchart of FIG. 7 may be viewed as depicting steps in a method implemented in the multi-function peripheral 103 in checking the user parameter table 169 for previously authenticated user parameters. Beginning with box 303, the Get Parameter method 179 examines the user parameter table 169 that contains any previously authenticated user parameters to determine if there is a match between the current unauthenticated user parameter and those authenticated user parameters listed in the table 169. If no match is detected in box 306, then the Get Parameter method 179 proceeds to box 309 in which in which a response is sent to the requesting authentication agent 156 that the user parameter has not been authenticated. Thereafter, the Get Parameter method 179 ends.

On the other hand, if a match is detected in box 306, then the Get Parameter method 179 proceeds to box 313 in which a response is sent to the requesting authentication agent that the user parameter has been authenticated. Thereafter, the Get Parameter method 179 ends.

With reference to FIG. 8, shown is a flow chart of the Set Parameter method 183 according to an aspect of the present invention. Alternatively, the flowchart of FIG. 8 may be viewed as depicting steps in a method implemented in the multi-function peripheral 103 in writing an authenticated user parameter to the user parameter table 169. In box 323, upon being called by an authentication agent 156 (FIG. 2) to write a user parameter to the user parameter table 169, then Set Parameter method 183 writes the user parameter to the user parameter table 169 for future reference. The user parameter may be communicated to the authentication manager 153 as an attribute in a call or in some other manner as can be appreciated by those with ordinary skill in the art. Thereafter, the Set Parameter method 183 ends.

With reference back to FIG. 2, the architecture of the authentication system 143 provides several advantages. One of these advantages includes the fact that additional authentication tasks may be added to the authentication system by simply adding an appropriate authentication agent 156 without making any change

to the authentication manager 153 or existing authentication agents 156, *etc.* Also, to create a new authentication agent 156, an existing authentication agent 156 may be copied and modified rather than creating the new authentication agent 156 from scratch. In addition redundant authentication of user parameters by different authentication agents 156 is avoided. Other advantages of the present invention will be apparent to one with ordinary skill in the art.

With reference back to FIG. 2, although the authentication system 143 of the present invention is embodied in software or code executed by general purpose hardware as discussed above, as an alternative the authentication system 143 may also be embodied in dedicated hardware or a combination of software/general purpose hardware and dedicated hardware. If embodied in dedicated hardware, the authentication system 143 can be implemented as a circuit or state machine that employs any one of or a combination of a number of technologies. These technologies may include, but are not limited to, discrete logic circuits having logic gates for implementing various logic functions upon an application of one or more data signals, application specific integrated circuits having appropriate logic gates, programmable gate arrays (PGA), field programmable gate arrays (FPGA), or other components, *etc.* Such technologies are generally well known by those skilled in the art and, consequently, are not described in detail herein.

The block diagrams and/or flow charts of FIGS. 2-8 show the architecture, functionality, and operation of an implementation of the authentication system 143. If embodied in software, each block may represent a module, segment, or portion of code that comprises program instructions to implement the specified logical function(s). The program instructions may be embodied in the form of source code that comprises human-readable statements written in a programming language or machine code that comprises numerical instructions recognizable by a suitable execution system such as a processor in a computer system or other system. The machine code may be converted from the source code, *etc.* If embodied in hardware, each block may represent a circuit or a number of interconnected circuits to implement the specified logical function(s).

Although the block diagrams and/or flow charts of FIGS. 2-8 show a specific order of execution, it is understood that the order of execution may differ from that which is depicted. For example, the order of execution of two or more blocks may be scrambled relative to the order shown. Also, two or more blocks shown in

succession in FIGS. 5-8 may be executed concurrently or with partial concurrence. In addition, any number of counters, state variables, warning semaphores, or messages might be added to the logical flow described herein, for purposes of enhanced utility, accounting, performance measurement, or providing troubleshooting aids, *etc.* It is understood that all such variations are within the scope of the present invention. Also, the block diagrams and/or flow charts of FIGS. 2-8 show are relatively self-explanatory and are understood by those with ordinary skill in the art to the extent that software and/or hardware can be created by one with ordinary skill in the art to carry out the various logical functions as described herein.

Also, where the authentication system 143 comprises software or code, it can be embodied in any computer-readable medium for use by or in connection with an instruction execution system such as, for example, a processor in a computer system or other system. In this sense, the logic may comprise, for example, statements including instructions and declarations that can be fetched from the computer-readable medium and executed by the instruction execution system. In the context of the present invention, a "computer-readable medium" can be any medium that can contain, store, or maintain the authentication system 143 for use by or in connection with the instruction execution system. The computer readable medium can comprise any one of many physical media such as, for example, electronic, magnetic, optical, electromagnetic, infrared, or semiconductor media. More specific examples of a suitable computer-readable medium would include, but are not limited to, magnetic tapes, magnetic floppy diskettes, magnetic hard drives, or compact discs. Also, the computer-readable medium may be a random access memory (RAM) including, for example, static random access memory (SRAM) and dynamic random access memory (DRAM), or magnetic random access memory (MRAM). In addition, the computer-readable medium may be a read-only memory (ROM), a programmable read-only memory (PROM), an erasable programmable read-only memory (EPROM), an electrically erasable programmable read-only memory (EEPROM), or other type of memory device.

Although the invention is shown and described with respect to certain preferred embodiments, it is obvious that equivalents and modifications will occur to others skilled in the art upon the reading and understanding of the specification.

The present invention includes all such equivalents and modifications, and is limited only by the scope of the claims.